Appl Intell (2014) 41:299–309
DOI 10.1007/s10489-014-0518-0

K-nearest neighbor-based weighted twin support vector regression

Yitian Xu · Laisheng Wang

Published online: 28 February 2014

Abstract Twin support vector regression (TSVR) finds ϵ-insensitive up- and down-bound functions by resolving a pair of smaller-sized quadratic programming problems (QPPs) rather than a single large one as in a classical SVR, which makes its computational speed greatly improved. However the local information among samples are not exploited in TSVR. To make full use of the knowledge of samples and improve the prediction accuracy, a K-nearest neighbor-based weighted TSVR (KNNWTSVR) is proposed in this paper, where the local information among samples are utilized. Specifically, a major weight is given to the training sample if it has more K-nearest neighbors. Otherwise a minor weight is given to it. Moreover, to further enhance the computational speed, successive overrelaxation approach is employed to resolve the QPPs. Experimental results on eight benchmark datasets and a real dataset demonstrate our weighted TSVR not only yields lower prediction error but also costs lower running time in comparison with other algorithms.

Keywords TSVR · K-nearest neighbor · Weights · Successive overrelaxation

Y. Xu (✉) · L. Wang
College of Science, China Agricultural University, Beijing 100083, China
e-mail: xytshuxue@126.com

Y. Xu
Department of Computer Science and Engineering, Arizona State University, Tempe, AZ 85287, USA

1 Introduction

The support vector machine (SVM) approach, motivated by VC dimensional theory and statistical learning theory [20], is a promising machine learning technique. Compared with other machine learning approaches like artificial neural networks [18], SVM gains many advantages. First, SVM solves a QPP, assuring that once an optimal solution is obtained, it is the unique (global) solution. Second, SVM derives a sparse and robust solution by maximizing the margin between the two classes. Third, SVM implements the structural risk minimization principle rather than the empirical risk minimization principle, which minimizes the upper bound of the generalization error [22]. At present, SVM has been successfully applied into various aspects ranging from machine learning, data mining to knowledge discovery [24].

However, one of the main challenges for the standard SVM is the high computational complexity. The training cost of $O(l^3)$, where l is the total size of training data, is expensive. To further improve the computational speed of SVM, Jayadeva et al. [8] proposed a twin support vector machine (TSVM) for the binary classification data in the spirit of the proximal SVM [5–7]. TSVM generates two nonparallel hyper-planes by solving two smaller-sized QPPs such that each hyper-plane is closer to one class and as far as possible from the other. The strategy of solving two smaller-sized QPPs rather than a single large one makes the learning speed of TSVM approximately four times faster than that of the standard SVM. At present, TSVM has become one of the popular methods because of its low computational complexity. Many variants of TSVM have been proposed in [9, 11, 17, 19, 23, 26]. Certainly, algorithms mentioned above are suitable for the classification problems. For the regression problem, Peng proposed an efficient TSVR [16].

It is well known that the same effects on the bound functions are considered to the training samples in TSVR.

However the local information among samples is ignored in TSVR. To enhance the prediction accuracy, Xu et al. proposed a weighted TSVR [25], where different penalty coefficients are given to the samples dissatisfying constraint depending on their different positions. In addition, Cheng et al. proposed a localized support vector machine (LSVM), where different penalty coefficients are given to the training samples dissatisfying the constraint depending on the similarities to the testing samples. Certainly the similarity is measured by K-nearest neighbor method. A point x is important if it has a larger number of K-nearest neighbors [3, 12, 28], whereas it is not important if it is an outlier. Motivated by the above studies, a K-nearest neighbor-based weighted TSVR (KNNWTSVR) is proposed in this paper [14], where different weights are proposed to give each training sample depending on their local information. Specifically, major weights are proposed to give the samples if they have more K-nearest neighbors. Whereas minor weights are given to the samples with a smaller number of K-nearest neighbors. Certainly our proposed algorithm only explored the relation among the training samples, and different weights are proposed to give each training sample, which is significantly different from LSVM. In addition, to further increase the computational speed of KNNWTSVR, a successive over relaxation method is employed to solve the QPPs, therefore we named it as SOR KNNWTSVR.

The effectiveness of the proposed algorithm is demonstrated by the experiments on eight benchmark datasets. The numerical results show that our proposed algorithms KNNWTSVR and SOR KNNWTSVR achieve nearly the same testing errors in all datasets. Moreover, they produce lower testing errors than SVR and TSVR for most cases. We can also find that SOR KNNWTSVR costs the least running time for almost all datasets. Finally we apply the new methods to predicting the content of protein by the spectral features of wheat. Compared with other algorithms, our proposed algorithms also yield lower prediction errors.

The paper is organized as follows. Section 2 outlines SVR and TSVR. A K-nearest neighbor-based weighted TSVR is proposed in Sect. 3, which includes the linear and nonlinear cases. A successive overrelaxation method for the KNNWTSVR is given in Sect. 4. Section 5 performs experiments on eight benchmark datasets and a real data experiment to investigate the effectiveness of the proposed algorithm. The last section contains the conclusions.

2 Related works

In this section, we give a brief description of the SVR and TSVR. Given a training set $T = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, where $x_i \in R^d$ and $y_i \in R$. For the sake of conciseness, let matrix $A = (x_1; x_2; \cdots; x_n)$ and matrix $Y = (y_1; y_2; \cdots; y_n)$.

2.1 Support vector regression

The nonlinear SVR seeks to find a regression function $f(x) = w^T\phi(x) + b$ in a high dimensional feature space tolerating the small error in fitting the given data points. This can be achieved by utilizing the ϵ-insensitive loss function that sets an ϵ-insensitive "tube" around the data, within which errors are discarded. The nonlinear SVR can be obtained by resolving the following QPP,

$$\begin{aligned} \min_{w,b,\xi,\xi^*} \quad & \frac{1}{2}\|w\|^2 + c\cdot\left(e^T\xi + e^T\xi^*\right) \qquad (1)\\ \text{s.t.} \quad & \left(\phi^T(A)w + eb\right) - Y \le e\epsilon + \xi,\\ & Y - \left(\phi^T(A)w + eb\right) \le e\epsilon + \xi^*,\\ & \xi \ge 0e, \quad \xi^* \ge 0e, \end{aligned}$$

where c is a parameter chosen a priori, which weights the tradeoff between the fitting error and flatness of the regression function, ξ and ξ^* are the slack vectors reflecting whether the samples locate into the ϵ-tube or not, e is the vector of ones of appropriate dimensions [25].

By introducing the Lagrangian multipliers α and α^*, we can derive the dual problem of the QPP (1) as follows,

$$\begin{aligned} \max_{\alpha,\alpha^*} \quad & -\frac{1}{2}(\alpha^* - \alpha)^T K\left(A, A^T\right)(\alpha^* - \alpha)\\ & + Y^T(\alpha^* - \alpha) + \epsilon e^T(\alpha^* + \alpha) \qquad (2)\\ \text{s.t.} \quad & e^T(\alpha^* - \alpha) = 0,\\ & 0e \le \alpha, \quad \alpha^* \le ce. \end{aligned}$$

Once the QPP (2) is resolved, we can achieve its solution $\alpha^{(*)} = (\alpha_1, \alpha_1^*, \alpha_2, \alpha_2^*, \ldots, \alpha_n, \alpha_n^*)$ and threshold b, and then obtain the regression function,

$$f(x) = \sum_{i=1}^{n}\left(\alpha_i^* - \alpha_i\right)K(x_i, x) + b. \qquad (3)$$

Here $K(x_i, x) = (\phi(x_i)\cdot\phi(x))$ represents a kernel function which gives the dot product in the high dimensional feature space.

2.2 Twin support vector regression

To further improve the computational speed, an efficient TSVM for the regression problem, termed as TSVR, was proposed in [16, 25]. TSVR generates an ϵ-insensitive down-bound function $f_1(x) = w_1^T x + b_1$ and an ϵ-insensitive up-bound function $f_2(x) = w_2^T x + b_2$. TSVR is illustrated in Fig. 1.

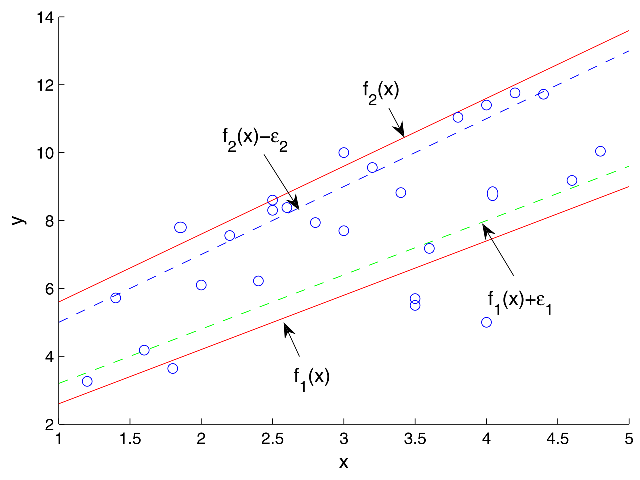

Fig. 1 Illustration of the TSVR

The final regressor $f(x)$ is decided by the mean of these two bound functions, i.e.,

$$\begin{aligned} f(x) &= \frac{1}{2}\big(f_1(x)+f_2(x)\big) \\ &= \frac{1}{2}(w_1+w_2)^T x + \frac{1}{2}(b_1+b_2). \end{aligned} \tag{4}$$

TSVR is obtained by solving the following pair of QPPs,

$$\begin{aligned} \min_{w_1,b_1,\xi} \quad & \frac{1}{2}\left\|Y - e\epsilon_1 - (Aw_1 + eb_1)\right\|^2 + c_1 e^T \xi \\ \text{s.t.} \quad & Y - (Aw_1+eb_1) \ge e\epsilon_1 - \xi, \\ & \xi \ge 0e, \end{aligned} \tag{5}$$

and

$$\begin{aligned} \min_{w_2,b_2,\eta} \quad & \frac{1}{2}\left\|Y + e\epsilon_2 - (Aw_2 + eb_2)\right\|^2 + c_2 e^T \eta \\ \text{s.t.} \quad & (Aw_2+eb_2) - Y \ge e\epsilon_2 - \eta, \\ & \eta \ge 0e, \end{aligned} \tag{6}$$

where c_1, c_2, ϵ_1 and ϵ_2 are parameters chosen a priori, ξ and η are slack vectors. By introducing the Lagrangian multipliers α and β, we derive their dual problems as follows,

$$\begin{aligned} \max_{\alpha} \quad & -\frac{1}{2}\alpha^T G\big(G^T G\big)^{-1} G^T \alpha \\ & + g^T G\big(G^T G\big)^{-1} G^T \alpha - g^T \alpha \\ \text{s.t.} \quad & 0e \le \alpha \le c_1 e, \end{aligned} \tag{7}$$

and

$$\begin{aligned} \max_{\beta} \quad & -\frac{1}{2}\beta^T G\big(G^T G\big)^{-1} G^T \beta \\ & - h^T G\big(G^T G\big)^{-1} G^T \beta + h^T \beta \\ \text{s.t.} \quad & 0e \le \beta \le c_2 e, \end{aligned} \tag{8}$$

where $G = [A\ e]$, $g = Y - e\epsilon_1$, and $h = Y + e\epsilon_2$.

Once the dual problems (7) and (8) are solved, we can get vectors w_1, b_1 and w_2, b_2 as follows,

$$\begin{bmatrix} w_1 \\ b_1 \end{bmatrix} = \big(G^T G\big)^{-1} G^T (g - \alpha), \tag{9}$$

and

$$\begin{bmatrix} w_2 \\ b_2 \end{bmatrix} = \big(G^T G\big)^{-1} G^T (h + \beta). \tag{10}$$

For the nonlinear case, TSVR resolves the following pair of QPPs,

$$\begin{aligned} \min_{w_1,b_1,\xi} \quad & \frac{1}{2}\left\|Y - e\epsilon_1 - \big(K\big(A, A^T\big)w_1 + eb_1\big)\right\|^2 + c_1 e^T \xi \\ \text{s.t.} \quad & Y - \big(K\big(A, A^T\big)w_1 + eb_1\big) \ge e\epsilon_1 - \xi, \\ & \xi \ge 0e, \end{aligned} \tag{11}$$

and

$$\begin{aligned} \min_{w_2,b_2,\eta} \quad & \frac{1}{2}\left\|Y + e\epsilon_2 - \big(K\big(A, A^T\big)w_2 + eb_2\big)\right\|^2 + c_2 e^T \eta \\ \text{s.t.} \quad & K\big(A, A^T\big)w_2 + eb_2 - Y \ge e\epsilon_2 - \eta, \\ & \eta \ge 0e. \end{aligned} \tag{12}$$

Similarly, we can derive the dual formulations of QPPs (11) and (12) as follows,

$$\begin{aligned} \max_{\alpha} \quad & -\frac{1}{2}\alpha^T H\big(H^T H\big)^{-1} H^T \alpha \\ & + g^T H\big(H^T H\big)^{-1} H^T \alpha - g^T \alpha \\ \text{s.t.} \quad & 0e \le \alpha \le c_1 e, \end{aligned} \tag{13}$$

and

$$\begin{aligned} \max_{\beta} \quad & -\frac{1}{2}\beta^T H\big(H^T H\big)^{-1} H^T \beta \\ & - h^T H\big(H^T H\big)^{-1} H^T \beta + h^T \beta \\ \text{s.t.} \quad & 0e \le \beta \le c_2 e, \end{aligned} \tag{14}$$

where $H = [K(A, A^T)\ e]$, $g = Y - e\epsilon_1$, and $h = Y + e\epsilon_2$.

Once the dual QPPs (13) and (14) are resolved, we can get vectors w_1, b_1 and w_2, b_2 as follows,

$$\begin{bmatrix} w_1 \\ b_1 \end{bmatrix} = \big(H^T H\big)^{-1} H^T (g - \alpha), \tag{15}$$

and

$$\begin{bmatrix} w_2 \\ b_2 \end{bmatrix} = \big(H^T H\big)^{-1} H^T (h + \beta). \tag{16}$$

Note that TSVR is comprised of a pair of QPPs such that each QPP determines one of up- or down-bound functions by using only one group of constraints compared with the standard SVR. Hence, TSVR resolves two smaller-sized QPPs rather than a single large one, which implies that TSVR works approximately four times faster than the standard SVR in theory. However, the information among samples are not exploited in TSVR, and some priori knowledge are ignored, which reduces the generalization performance of TSVR to a certain degree.

3 K-nearest neighbor-based weighted twin support vector regression

In this section, we first present the concept of K-nearest neighbor [12], and then propose a K-nearest neighbor-based weighted TSVR. The weight of each training sample varies according to its local information.

3.1 K-nearest neighbor method

Given l data points $x_1, x_2, \dots, x_l$, where $x_i \in R^n$ from the underlying submanifold M, one can build a nearest neighbor graph G to model the local geometrical structure of M. For each data point x_i, we find its k nearest neighbors and put an edge between x_i and its neighbors. Let $N(x_i) = \{x_i^1, x_i^2, \dots, x_i^k\}$ be the set of its K-nearest neighbors. Thus, the weight matrix of G can be defined as follows:

$$W_{i,j} = \begin{cases} 1, & \text{if } x_i \text{ is } k\text{-nearest neighbors of } x_j \\ & \quad \text{or } x_j \text{ is } k\text{-nearest neighbors of } x_i, \\ 0, & \text{otherwise.} \end{cases} \tag{17}$$

The nearest neighbor graph G with weight matrix W characterizes the local geometry of the data manifold. It has been frequently used in manifold based learning techniques.

To embody the importance of a point, we introduce a new variable $d_i = \sum_{j=1}^{l} W_{ij}$ $(i = 1, 2, \dots, l)$ for each sample. A larger value d_i denotes the i-th sample possesses more neighbors, which implies that the i-th sample is important. According to this principle, we present the following weighted algorithm.

3.2 Linear case

In TSVR, all samples are equally weighted. This implies that samples will play the same roles on the bound functions no matter whether they are important or not. Before presenting the new algorithm, we first modify the original TSVR, and it is illustrated in Fig. 2. As we can learn that it degrades into the original SVR when $f_1(x) = f_2(x)$. It is well known that

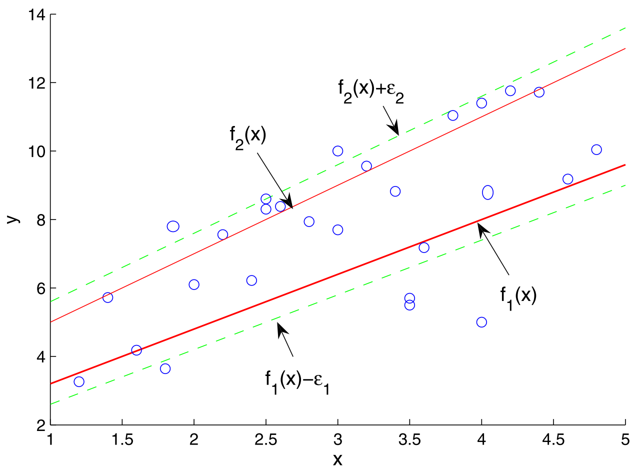

Fig. 2 Illustration of the modified TSVR

different samples have different effects on the bound functions. A point x is more important if it owns a great number of K-nearest neighbors. Motivated by the above studies, a K-nearest neighbor-based weighted TSVR is proposed in this section, where different weights are proposed to give each sample depending on their local information.

$$\begin{aligned} \min_{w_1, b_1, \xi} \quad & \frac{1}{2}\big(Y - (Aw_1 + eb_1)\big)^T D\big(Y - (Aw_1 + eb_1)\big) \\ & + c_1 e^T \xi \\ \text{s.t.} \quad & Y - (Aw_1 + eb_1) \geq -\epsilon_1 e - \xi, \\ & \xi \geq 0, \end{aligned} \tag{18}$$

and

$$\begin{aligned} \min_{w_2, b_2, \eta} \quad & \frac{1}{2}\big(Y - (Aw_2 + eb_2)\big)^T D\big(Y - (Aw_2 + eb_2)\big) \\ & + c_2 e^T \eta \\ \text{s.t.} \quad & (Aw_2 + eb_2) - Y \geq -\epsilon_2 e - \eta, \\ & \eta \geq 0, \end{aligned} \tag{19}$$

where $c_1, c_2 > 0$, $\varepsilon_1, \varepsilon_2 \geq 0$ are parameters chosen a priori. $D = diag(d_1, d_2, \dots, d_l)$ is a weighted diagonal matrix, and $d_i = \sum_{j=1}^{l} W_{ij}$. The larger value d_i is, the more important the i-th sample is. Then it is more reasonable to give it a major weight.

Note that there are two terms in the objective function of (18). The first term is the sum of squared distances from the shifted function $f_1(x)$ to the training points. Moreover different weights are given to each training sample depending on their local information. This is significantly different from the LSVM. In the second term, slack vector ξ_i is introduced to measure the error whether the distance from the down bound function to the sample is smaller than ϵ_1. However, the second term in the objective function of the LSVM [2] means that different penalties are given to the training samples depending on the similarity to the test examples. The similarity is captured by a weighted function σ. Certainly the weighted penalties are only given to the training samples dissatisfying the constraint.

To resolve (18), we first introduce the following Lagrangian function,

$$\begin{aligned} L = {} & \frac{1}{2}\big(Y-(Aw_1+eb_1)\big)^T D\big(Y-(Aw_1+eb_1)\big)+c_1e^T\xi \\ & -\alpha^T\big(Y-(Aw_1+eb_1)+\epsilon_1 e+\xi\big)-\beta^T\xi, \end{aligned} \tag{20}$$

where $\alpha, \beta \geq 0$ are Lagrangian multipliers. Differentiating the Lagrangian function with respect to w_1, b_1 and ξ yields the following Karush-Kuhn-Tucker (KKT) conditions:

$$\frac{\partial L}{\partial w_1} = -A^T D(Y-Aw_1-eb_1)+A^T\alpha = 0, \tag{21}$$

$$\frac{\partial L}{\partial b_1} = -e^T D(Y-Aw_1-eb_1)+e^T\alpha = 0, \tag{22}$$

$$\frac{\partial L}{\partial \xi} = c_1 e-\alpha-\beta = 0, \tag{23}$$

$$\alpha^T\big(Y-(Aw_1+eb_1)+\epsilon_1 e+\xi\big) = 0, \tag{24}$$

$$\beta^T\xi = 0. \tag{25}$$

Combining (21) and (22), we can achieve

$$-\begin{bmatrix} A^T \\ e^T \end{bmatrix} D(Y-Aw_1-eb_1)+\begin{bmatrix} A^T \\ e^T \end{bmatrix}\alpha = 0. \tag{26}$$

Define

$$G = [A \quad e], \qquad u_1 = \begin{bmatrix} w_1^T & b_1 \end{bmatrix}^T, \tag{27}$$

then we have

$$-G^T D(Y-Gu_1)+G^T\alpha = 0, \tag{28}$$

i.e.

$$u_1 = \big(G^T DG\big)^{-1}G^T(DY-\alpha). \tag{29}$$

Finally, the dual formulation of (18) can be derived as

$$\begin{aligned} \max_{\alpha} \quad & -\frac{1}{2}\alpha^T G\big(G^T DG\big)^{-1}G^T\alpha - Y^T\alpha-\epsilon_1 e^T\alpha \\ & + Y^T DG\big(G^T DG\big)^{-1}G^T\alpha+\frac{1}{2}Y^T DY \\ & -\frac{1}{2}Y^T DG\big(G^T DG\big)^{-1}G^T DY \\ \text{s.t.} \quad & 0 \leq \alpha \leq c_1 e, \end{aligned} \tag{30}$$

where $\frac{1}{2}Y^T DY - \frac{1}{2}Y^T DG(G^T DG)^{-1}G^T DY$ is a constant for the variable α, we may discard it and resolve the following simplified QPP,

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2}\alpha^T G\big(G^T DG\big)^{-1}G^T\alpha + Y^T\alpha+\epsilon_1 e^T\alpha \\ & - Y^T DG\big(G^T DG\big)^{-1}G^T\alpha \\ \text{s.t.} \quad & 0 \leq \alpha \leq c_1 e. \end{aligned} \tag{31}$$

Similarly, the dual formulation of (19) can be derived as follows,

$$\begin{aligned} \min_{\gamma} \quad & \frac{1}{2}\gamma^T G\big(G^T DG\big)^{-1}G^T\gamma + \epsilon_2 e^T\gamma \\ & + Y^T DG\big(G^T DG\big)^{-1}G^T\gamma - Y^T\gamma \\ \text{s.t.} \quad & 0 \leq \gamma \leq c_2 e. \end{aligned} \tag{32}$$

Notice that $G^T DG$ is always positive semidefinite, it is possible that it may not be well conditioned in some situations. To overcome this ill-conditioning case, we introduce a regularization term δI, where δ is a very small positive number, such as $\delta = 1e-7$. Then (29) is modified to

$$\mu_1 = \big(G^T DG+\sigma I\big)^{-1}G^T(DY-\alpha). \tag{33}$$

Similarly, μ_2 is modified to

$$\mu_2 = \begin{bmatrix} w_2^T & b_2 \end{bmatrix}^T = \big(G^T DG+\sigma I\big)^{-1}G^T(DY+\gamma). \tag{34}$$

Once the vectors μ_1 and μ_2 are known from (33) and (34), the up- and down-bound functions can be obtained. The end regressor is decided by the mean of these two functions,

$$\begin{aligned} f(x) &= \frac{1}{2}\big(f_1(x)+f_2(x)\big) \\ &= \frac{1}{2}(w_1+w_2)^T x+\frac{1}{2}(b_1+b_2). \end{aligned} \tag{35}$$

3.3 Nonlinear case

In this subsection, we extend the linear KNNWTSVR to the nonlinear case using the kernel trick. The input data are mapped into a high dimensional feature space by some nonlinear kernel functions. In the feature space, a linear regression function is implemented which corresponds to a nonlinear regression function in the input space [10],

$$\begin{aligned} \min_{w_1,b_1,\xi} \quad & \frac{1}{2}\big(Y-\big(K\big(A,A^T\big)w_1+eb_1\big)\big)^T \\ & \times D\big(Y-\big(K\big(A,A^T\big)w_1+eb_1\big)\big)+c_1e^T\xi \\ \text{s.t.} \quad & Y-\big(K\big(A,A^T\big)w_1+eb_1\big) \geq -\epsilon_1 e-\xi, \\ & \xi \geq 0, \end{aligned} \tag{36}$$

and

$$\begin{aligned}\min_{w_2,b_2,\eta}\quad & \frac{1}{2}\big(Y-\big(K\big(A,A^T\big)w_2+eb_2\big)\big)^T \\ & \times D\big(Y-\big(K\big(A,A^T\big)w_2+eb_2\big)\big)+c_2e^T\eta \quad (37)\\ \text{s.t.}\quad & \big(K\big(A,A^T\big)w_2+eb_2\big)-Y\geq-\epsilon_2e-\eta,\\ & \eta\geq 0,\end{aligned}$$

where $c_1, c_2 > 0$ and $\varepsilon_1, \varepsilon_2 \geq 0$ are parameters chosen a priori. $D = diag(d_1, d_2, \dots, d_l)$ is a weighted diagonal matrix, and $d_i = \sum_{j=1}^{l} W_{ij}$. To resolve the QPP (36), we first introduce the following Lagrangian function,

$$\begin{aligned}L=&\frac{1}{2}\big(Y-\big(K\big(A,A^T\big)w_1+eb_1\big)\big)^T\\ &\times D\big(Y-\big(K\big(A,A^T\big)w_1+eb_1\big)\big)+c_1e^T\xi\\ &-\alpha^T\big(Y-\big(K\big(A,A^T\big)w_1+eb_1\big)+\epsilon_1e+\xi\big)-\beta^T\xi,\end{aligned}\quad (38)$$

where $\alpha, \beta \geq 0$ are Lagrangian multipliers. Differentiating the Lagrangian function with respect to w_1, b_1 and ξ yields the following KKT conditions,

$$\begin{aligned}\frac{\partial L}{\partial w_1}=&-K\big(A,A^T\big)^TD\big(Y-K\big(A,A^T\big)w_1-eb_1\big)\\ &+K\big(A,A^T\big)^T\alpha=0,\end{aligned}\quad (39)$$

$$\frac{\partial L}{\partial b_1}=-e^TD\big(Y-K\big(A,A^T\big)w_1-eb_1\big)+e^T\alpha=0,\quad (40)$$

$$\frac{\partial L}{\partial \xi}=c_1e-\alpha-\beta=0,\quad (41)$$

$$\alpha^T\big(Y-\big(K\big(A,A^T\big)w_1+eb_1\big)+\epsilon_1e+\xi\big)=0,\quad (42)$$

$$\beta^T\xi=0.\quad (43)$$

Combining Eqs. (39) and (40), we get

$$\begin{aligned}&-\begin{bmatrix}K(A,A^T)^T\\ e^T\end{bmatrix}D\big(Y-K\big(A,A^T\big)w_1-eb_1\big)\\ &+\begin{bmatrix}K(A,A^T)^T\\ e^T\end{bmatrix}\alpha=0.\end{aligned}\quad (44)$$

Define

$$H=\big[K\big(A,A^T\big)\quad e\big]\quad\text{and}\quad u_1=\big[w_1^T\quad b_1\big]^T,\quad (45)$$

we can then get

$$-H^TD(Y-Hu_1)+H^T\alpha=0.\quad (46)$$

We further get

$$u_1=\big(H^TDH\big)^{-1}H^T(DY-\alpha).\quad (47)$$

Finally, the dual formulation of (36) can be derived as follows,

$$\begin{aligned}\max_{\alpha}\quad & -\frac{1}{2}\alpha^TH\big(H^TDG\big)^{-1}H^T\alpha-Y^T\alpha-\epsilon_1e^T\alpha\\ & +Y^TDH\big(H^TDH\big)^{-1}H^T\alpha+\frac{1}{2}Y^TDY\\ & -\frac{1}{2}Y^TDH\big(H^TDH\big)^{-1}H^TDY \quad (48)\\ \text{s.t.}\quad & 0\leq\alpha\leq c_1e.\end{aligned}$$

Since $\frac{1}{2}Y^TDY-\frac{1}{2}Y^TDH(H^TDH)^{-1}H^TDY$ is a constant for the variable α, we can discard it and resolve the following dual formulation,

$$\begin{aligned}\min_{\alpha}\quad & \frac{1}{2}\alpha^TH\big(H^TDH\big)^{-1}H^T\alpha+Y^T\alpha+\epsilon_1e^T\alpha\\ & -Y^TDH\big(H^TDH\big)^{-1}H^T\alpha \quad (49)\\ \text{s.t.}\quad & 0\leq\alpha\leq c_1e.\end{aligned}$$

Similarly, we can derive the following dual formulation of (37) as follows,

$$\begin{aligned}\min_{\gamma}\quad & \frac{1}{2}\gamma^TH\big(H^TDH\big)^{-1}H^T\gamma+\epsilon_2e^T\gamma\\ & +Y^TDH\big(H^TDH\big)^{-1}H^T\gamma-Y^T\gamma \quad (50)\\ \text{s.t.}\quad & 0\leq\gamma\leq c_2e.\end{aligned}$$

Once the dual QPPs (49) and (50) are solved, we can get vectors μ_1 and μ_2 for the ill-condition case,

$$\mu_1=\big(H^TDH+\sigma I\big)^{-1}H^T(DY-\alpha),\quad (51)$$

and

$$\mu_2=\big[w_2^T\quad b_2\big]^T=\big(H^TDH+\sigma I\big)^{-1}H^T(DY+\gamma).\quad (52)$$

Finally, we can achieve the end regressor for the nonlinear case as follows,

$$\begin{aligned}f(x)&=\frac{1}{2}\big(f_1(x)+f_2(x)\big)\\ &=\frac{1}{2}(w_1+w_2)^TK(A,x)+\frac{1}{2}(b_1+b_2).\end{aligned}\quad (53)$$

3.4 Computational complexity analysis

The computational complexity is discussed in this subsection. Suppose there are l training samples. From the QPP (2) we can know that SVR resolves a larger-sized QPP with two inequality constraints and an equality constraint. However TSVR resolves a pair of smaller-sized QPPs, each of which has only one inequality constraint. So TSVR works approximately 4 times faster than SVR in theory. In the following we discuss the computational complexity of our proposed KNNWTSVR. The main computational process in our algorithm involves two steps:

(1) K-nearest neighbors graph. Using $O(l^2 \log(l))$ for all the l training points to compute the weight matrix W [1].

(2) Optimization of our KNNWTSVR. KNNWTSVR has the similar dual formulation to the TSVR, then KNNWTSVR spends nearly the same running time as TSVR does. It means that the KNNWTSVR does not improve the computational complexity.

4 Successive overrelaxation approach

In this subsection, we discuss the implementation of KNNWTSVR. Successive overrelaxation is an iterative procedure that employs the Gauss-Seidel iterations with the extrapolation factor $t \in (0, 2)$ to accelerate the solving of the QPPs with linear convergence [13].

We can easily rewrite QPPs (31), (32), (49) and (50) into the following unified form,

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2}\alpha^T Q\alpha + d\alpha \\ \text{s.t.} \quad & \alpha \in S = \{0 \le \alpha \le ce\}. \end{aligned} \tag{54}$$

We take the dual formulation (31) for example, it can be rewritten in the above unified form, where $Q = G(G^T DG + \sigma I)^{-1}G^T$ and $d = Y^T + \epsilon_1 e^T - Y^T DQ$.

A necessary and sufficient optimality condition for (31) is the following gradient projection optimality condition,

$$\alpha = \big(\alpha - t_1 D_1^{-1}(Q\alpha + Y + \epsilon_1 e - QDY)\big)_*, \tag{55}$$

where $(\cdot)_*$ denotes the 2-norm projection on the feasible region S, that is

$$\big((\alpha)_*\big)_i = \begin{cases} 0, & \alpha_i \le 0, \\ \alpha_i, & 0 \le \alpha_i \le c_1, \\ c_1, & \alpha_i \ge c_1. \end{cases} \tag{56}$$

Define $Q = L + D_1 + L^T$, where $L \in R^{l\times l}$ constitutes the strictly lower triangular part of the symmetric matrix of Q, and the nonzero elements of D_1 constitute the diagonal of Q. We can split the matrix Q into the sum of two matrices B and C as follows:

$$Q = t_1^{-1} D_1(B + C), \tag{57}$$

where

$$B = I + t_1 D_1^{-1} L, \tag{58}$$

$$C = (t_1 - 1)I + t_1 D_1^{-1} L^T, \tag{59}$$

where $0 < t_1 < 2$, I is an identity matrix of appreciate dimensions. The matrix $B + C$ is positive semidefinite, and matrix $B - C$ is positive definite. Finally we can derive the following iterative formula,

$$\alpha_{r+1} = \big(\alpha_{r+1} - B\alpha_{r+1} - C\alpha_r + t_1 D_1^{-1}(Y + \epsilon_1 e - QDY)\big)_*. \tag{60}$$

Similarly, we can obtain the other iterative formula,

$$\gamma_{r+1} = \big(\gamma_{r+1} - B\gamma_{r+1} - C\gamma_r + t_2 D_1^{-1}(\epsilon_2 e + QDY - Y)\big)_*. \tag{61}$$

The flowchart of SOR KNNWTSVR is described as follows:

SOR KNNWTSVR algorithm

Input: The matrix Q and vector d.

1. Select parameters $t_1, t_2 \in (0, 2)$, and initialize α_0, $\gamma_0 \in R^l$;

2. Compute α and γ as following formulation $\alpha_{r+1} = (\alpha_r - t_1 D_1^{-1}(Q\alpha + Y + \epsilon_1 e - QDY + L(\alpha_{r+1} - \alpha_r)))_*$ and $\gamma_{r+1} = (\gamma_r - t_2 D_1^{-1}(Q\gamma - Y + \epsilon_2 e + QDY + L(\gamma_{r+1} - \gamma_r)))_*$. Where $Q = G(G^T DG + \sigma I)^{-1}G^T$. Define $Q = L + D_1 + L^T$, where $L \in R^{l\times l}$ is a strictly lower triangular matrix, and $D_1 \in R^{l\times l}$ is a diagonal matrix.

3. Compute $\|\alpha_{r+1} - \alpha_r\|$ and $\|\gamma_{r+1} - \gamma_r\|$ until they are less than some prescribed tolerance.

Output: the optimal values of α and γ.

It is well known that the successive overrelaxation method is linear convergence, but other iterative methods, e.g., Newton iteration, and so on, are quadratic convergence. So the successive overrelaxation method is employed to accelerate the speed of KNNWTSVR.

5 Numerical experiments

We conduct two kinds of experiments including eight benchmark datasets from the UCI machine learning repository[1] and a real dataset to investigate the effectiveness of our KNNWTSVR. All experiments are carried out in Matlab 7.9 (R2009b) on Windows XP running on a PC with system configuration Intel(R) Core(TM) 2 Duo CPU E7500 (2.93 GHz) with 3.00 GB of RAM.

5.1 Evaluation criteria

To evaluate the performance of our proposed algorithm, the evaluation criteria are specified before presenting the experimental results. The total number of testing samples is denoted by m, while y_i denotes the real-value of a sample point

[1] http://archive.ics.uci.edu/ml/datasets.html.

x_i, $\hat{y}_i$ denotes the predicted value of x_i, and $\bar{y} = \sum_{i=1}^{m} y_i$ is the mean of $y_1, y_2, \ldots, y_m$. We use the following criteria for algorithm evaluation [16, 22].

MAE: Mean absolute error, defined as

$$MAE = \frac{1}{m}\sum_{i=1}^{m} |y_i - \hat{y}_i|. \tag{62}$$

MAE is also a popular deviation measurement between the real and predicted values.

RMSE: Root mean squared error, defined as

$$RMSE = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2}. \tag{63}$$

SSE/SST: Ratio between sum of squared error and sum of squared deviation of testing samples, defined as

$$SSE/SST = \sum_{i=1}^{m}(y_i - \hat{y}_i)^2 \Big/ \sum_{i=1}^{m}(y_i - \bar{y}_i)^2. \tag{64}$$

SSR/SSE: Ratio between interpretable sum of squared deviation and real sum of squared deviation of testing samples, defined as

$$SSR/SSE = \sum_{i=1}^{m}(\hat{y}_i - \bar{y}_i)^2 \Big/ \sum_{i=1}^{m}(y_i - \hat{y}_i)^2. \tag{65}$$

In most cases, small SSE/SST means there is good agreement between the estimates and the real values, and decreasing SSE/SST is usually accompanied by an increase in SSR/SST. However, an extremely small value for SSE/SST is in fact not good, for it probably means that the regressor is over-fitting the data.

5.2 Parameters selection

We know that the kernel function and its parameters in SVM have great effects on the experimental results [4]. In our experiments, we only consider the Gaussian kernel function $k(x_i, x_j) = \exp(-\|x_i - x_j\|^2/\gamma^2)$ for these datasets as it is often employed and yields great generalization performance. We choose optimal values for the parameters by the grid search method. The optimal Gaussian kernel parameter γ was selected over the range $\{2^i \mid i = -3, -2, \ldots, 9\}$. The optimal values of parameter c in all algorithms were searched from set $\{2^i \mid i = -3, -2, \ldots, 8\}$. The parameter ε in all algorithms was searched from the set $\{\frac{i}{10} \mid i = 1, 2, \ldots, 9\}$. The parameter t in our proposed algorithm ranged from the set $\{0.25, 0.5, 0.75, 1, 1.25, 1.5, 1.75\}$.

5.3 Experiments on benchmark datasets

In this section, we performed experiments on eight benchmark datasets, they are Diabetes, Pyrim, Con.s, Machine cpu, Triazines, Auto-mpg, Auto-price, and Chwirut. For each experiment, we use 5-fold cross-validation to evaluate the performance of four algorithms, i.e., KNNWTSVR, SOR KNNWTSVR, TSVR and SVR. That is to say, the dataset is split randomly into five subsets, and one of those sets is reserved as a test set. This process is repeated five times, and the average value of five testing results is used as the performance measure. We test the validity of the proposed algorithm from both testing errors and computational time aspects.

The performance comparisons of four algorithms are summed in Table 1. In error items, the first item denotes the mean value of five times testing results, and the second item stands for plus or minus the standard deviation. Time denotes the mean value of time taken by five experiments, and each experimental time consists of training time and testing time.

From the perspective of prediction error, we can find that KNNWTSVR and SOR KNNWTSVR yield nearly the same testing errors on eight benchmark datasets. Moreover they are lower than that produced by SVR and TSVR for most cases.

In terms of computational time, our SOR KNNWTSVR costs the least running time among four algorithms for most cases. The next is our KNNWTSVR. However SVR costs the most computational time. The main reason lies in that SVR resolves a larger-sized QPP, but three other algorithms resolves a pair of smaller-sized QPP. Although the TSVR works four times faster than SVR in theory, sometimes more than four times, as we can learn from the datasets Machine cpu, Triazines, Auto-mpg, Auto-pric, and Chwirut. The possible reason is that there is an equality constraint in the dual formulation (2) of SVR, but only an inequality constraint in the dual formulation of TSVR, our KNNWTSVR, and SOR KNNWTSVR.

Generally speaking, as the size of the training dataset increases, our SOR KNNWTSVR has more obvious advantages in computational time.

5.4 Experiment on a real dataset

In this real data experiment, there are 210 wheat samples from all over China. The protein content of wheat ranges from 9.83 % to 20.26 %. They are provided by Heilongjiang research institute of agricultural science. Each sample has 1193 spectral features. They were scanned in transmission mode using a commercial spectrometer MATRIX-I. Samples were acquired in a rectangular quartz cuvette of 1-mm path length with air as reference at room temperature (20–24 °C). The reference spectrum was subtracted from the

Table 1 Performance comparisons of four algorithms with Gaussian kernel function on eight benchmark datasets

Datasets	Algorithm	MAE	RMSE	SSE/SST	SSR/SST	Time
Diabetes (43 × 3)	SVR	0.458 ± 0.108	0.560 ± 0.081	0.800 ± 0.215	0.515 ± 0.241	0.142
	TSVR	0.466 ± 0.093	0.553 ± 0.072	0.792 ± 0.277	0.650 ± 0.215	0.061
	KNNWTSVR	0.460 ± 0.072	0.552 ± 0.044	0.793 ± 0.272	0.322 ± 0.318	0.039
	SOR KNNWTSVR	**0.427 ± 0.064**	0.518 ± 0.069	0.706 ± 0.266	0.288 ± 0.189	**0.025**
Pyrim (74 × 28)	SVR	0.050 ± 0.020	0.072 ± 0.041	0.999 ± 0.538	1.157 ± 0.708	1.175
	TSVR	**0.049 ± 0.012**	0.066 ± 0.025	0.527 ± 0.321	0.984 ± 0.549	0.102
	KNNWTSVR	**0.049 ± 0.014**	0.069 ± 0.032	0.707 ± 0.629	0.639 ± 0.554	0.097
	SOR KNNWTSVR	**0.049 ± 0.014**	0.069 ± 0.030	0.667 ± 0.402	0.637 ± 0.555	**0.067**
Con.s (103 × 7)	SVR	0.216 ± 0.032	0.253 ± 0.031	0.781 ± 0.232	0.257 ± 0.100	1.218
	TSVR	0.182 ± 0.027	0.221 ± 0.022	0.620 ± 0.210	0.500 ± 0.258	0.153
	KNNWTSVR	**0.177 ± 0.023**	0.216 ± 0.019	0.592 ± 0.210	0.060 ± 0.064	0.143
	SOR KNNWTSVR	**0.177 ± 0.020**	0.219 ± 0.020	0.611 ± 0.202	0.060 ± 0.064	**0.130**
Machine cpu (209 × 6)	SVR	89.262 ± 22.047	159.82 ± 68.629	1.542 ± 0.791	0.542 ± 0.791	18.850
	TSVR	**91.248 ± 20.379**	144.844 ± 60.122	1.381 ± 0.938	0.509 ± 0.864	1.006
	KNNWTSVR	93.852 ± 17.467	146.766 ± 59.093	1.412 ± 0.923	0.453 ± 0.816	0.549
	SOR KNNWTSVR	93.436 ± 17.502	146.766 ± 59.097	1.409 ± 0.915	0.449 ± 0.808	**0.537**
Triazines (186 × 60)	SVR	0.112 ± 0.015	0.153 ± 0.013	0.998 ± 0.087	0.171 ± 0.126	8.345
	TSVR	0.131 ± 0.008	0.144 ± 0.011	0.885 ± 0.064	0.193 ± 0.076	0.469
	KNNWTSVR	**0.104 ± 0.008**	0.143 ± 0.006	0.880 ± 0.125	0.135 ± 0.126	0.462
	SOR KNNWTSVR	**0.104 ± 0.007**	0.144 ± 0.008	0.884 ± 0.112	0.132 ± 0.120	**0.446**
Auto-mpg (392 × 7)	SVR	6.879 ± 0.704	8.801 ± 0.714	1.483 ± 0.531	0.483 ± 0.531	396.6
	TSVR	5.004 ± 1.270	8.887 ± 2.898	0.619 ± 0.347	1.298 ± 0.538	**1.928**
	KNNWTSVR	**3.995 ± 1.450**	7.977 ± 1.237	1.381 ± 0.340	0.430 ± 0.330	2.170
	SOR KNNWTSVR	3.997 ± 1.445	7.977 ± 1.237	1.381 ± 0.340	0.430 ± 0.330	2.018
Auto-price (159 × 15)	SVR	5103.0 ± 2147.9	6713.3 ± 2534.7	1.6788 ± 0.894	0.678 ± 0.894	5.175
	TSVR	**4433.0 ± 2098.0**	6444.4 ± 2892.1	2.281 ± 2.537	2.214 ± 2.237	0.341
	KNNWTSVR	4644.5 ± 1178.5	5622.1 ± 1687.6	1.594 ± 0.947	0.595 ± 0.947	0.329
	SOR KNNWTSVR	4644.5 ± 1178.5	5622.1 ± 1687.7	1.594 ± 0.947	0.595 ± 0.947	**0.303**
Chwirut (214 × 1)	SVR	0.208 ± 0.033	0.283 ± 0.041	1.170 ± 0.150	0.170 ± 0.150	22.056
	TSVR	0.029 ± 0.010	0.040 ± 0.014	1.147 ± 0.651	1.033 ± 0.137	1.859
	KNNWTSVR	0.028 ± 0.008	0.038 ± 0.015	0.024 ± 0.013	0.901 ± 0.148	0.777
	SOR KNNWTSVR	**0.023 ± 0.010**	0.034 ± 0.012	0.020 ± 0.015	0.840 ± 0.300	**0.755**

Table 2 Performance comparisons of four algorithms with Gaussian kernel function on wheat dataset

Algorithms	MAE	RMSE	SSE/SST	SSR/SST	Time (s)	Optimal parameters
SVR	1.171 ± 0.327	1.492 ± 0.414	1.071 ± 0.063	0.071 ± 0.063	21.885	$(C = 16, \epsilon = 0.9, \gamma = 0.25)$
TSVR	0.955 ± 0.187	1.199 ± 0.251	0.794 ± 0.442	0.498 ± 0.172	1.428	$(C = 8, \epsilon = 0.9, \gamma = 8)$
KNNWTSVR	0.866 ± 0.250	1.080 ± 0.315	0.732 ± 0.614	0.611 ± 0.169	1.451	$(C = 1, \epsilon = 0.3, \gamma = 32)$
SOR KNNWTSVR	**0.799 ± 0.230**	**1.014 ± 0.293**	0.665 ± 0.571	0.665 ± 0.163	**1.353**	$(C = 32, \epsilon = 0.9, \gamma = 32, t = 1)$

sample spectra to remove background noise. The rectangular quartz cuvette was cleaned after each sample was scanned to minimize cross-contamination [27].

In this high dimensional data experiment, we predict the content of protein using 1193 spectral features of wheat. The prediction errors of four algorithms are summed in Table 2.

From Table 2, we can learn that SOR KNNWTSVR yields lowest prediction error (0.799 %) among four algorithms. The prediction error (0.866 %) of KNNWTSVR follows. The errors produced by them are lower than that produced by SVR (1.171 %) and TSVR (0.955 %). The main reason is that SOR KNNWTSVR and KNNWTSVR consider the local information among samples, and different weights are proposed to give the samples depending on their local information. But in SVR and TSVR, all samples are equally weighted. In addition, to further improve the prediction accuracy, we can incorporate feature selection into our proposed algorithms. The first step is to reduce the dimensions of training samples [15, 21], and then predict the content of protein by the reduced spectral features of wheat.

In terms of computational time, SVR costs the most running time (21.885 s) among four algorithms. TSVR and KNNWTSVR cost nearly the same computational time, and they are slightly higher than that of SOR KNNWTSVR. Moreover, we can further find that TSVR, KNNWTSVR and SOR KNNWTSVR work more than four times faster than SVR. The possible reason is that only one inequality constraint is included in the dual formulations of them, but two constraints including an equality and an inequality constraints in SVR.

6 Conclusion

A K-nearest neighbor-based weighted TSVR is proposed in this paper, and the local information among samples is exploited in the model. The KNN method is employed to count the number d_i of neighbors for each sample. The larger value d_i denotes the i-th sample has a great number of K-nearest neighbors, and more important for the i-th sample. So a major weight is proposed to give the i-th sample. In contrast, a minor weight is given to the samples with a smaller number of k-nearest neighbors. Moreover successive overrelaxation method is used to further improve the speed of KNNWTSVR. Experimental results on eight benchmark datasets and a real dataset demonstrate the validity of our proposed algorithms. Our SOR KNNWTSVR not only yields lower testing error but also costs the least running time. How to incorporate the feature selection into our proposed algorithms when addressing the high dimensional data is our future research.

Acknowledgements The authors gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation. This work was supported by the National Natural Science Foundation of China (No. 61153003, 11171346 and 11271367) and China Scholarship Fund (No. 201208110282).

References

1. Cai D, He X, Zhang W, Han J (2007) Regularized locality preserving indexing via spectral regression. In: Proc 2007 ACM int. conf. on information and knowledge management
2. Cheng H, Tan P, Jin R (2010) Efficient algorithm for localized support vector machine. IEEE Trans Knowl Data Eng 22(4):537–549
3. Cover T, Hart P (1967) Nearest neighbor pattern classification. IEEE Trans Inf Theory 13:21–27
4. Diosan L, Rogozan A, Pecuchet J-P (2013) Improving classification performance of support vector machine by genetically optimising kernel shape and hyper-parameters. Appl Intell 36:280–294
5. Fung G, Mangasarian O (2001) Proximal support vector machine classifiers. In: Seven international proceedings on knowledge discovery and data mining, pp 77–86
6. Fung G, Mangasarian O (2005) Multicategory proximal support vector machine classifiers. Mach Learn 59:77–97
7. Ghorai S, Mukherjee A, Dutta P (2009) Nonparallel plane proximal classifier. Signal Process 89(4):510–522
8. Jayadeva, Khemchandani R, Chandra S (2007) Twin support vector machines for pattern classification. IEEE Trans Pattern Anal Mach Intell 29(5):905–910
9. Jayadeva, Khemchandani R, Chandra S (2007) Fuzzy multi-category proximal support vector classification via generalized eigenvalues. Soft Comput 11(7):679–685
10. Khemchandani R, Jayadeva, Chandra S (2009) Optimal kernel selection in twin support vector machines. Optim Lett 3(1):77–88
11. Kumar M, Gopal M (2009) Least squares twin support vector machines for pattern classification. Expert Syst Appl 36(4):7535–7543
12. Li I, Chen J, Wu J (2013) A fast prototype reduction method based on template reduction and visualization-induced self-organizing map for nearest neighbor algorithm. Appl Intell 39:564–582
13. Mangasarian O, Musicant D (1999) Successive overrelaxation for support vector machines. IEEE Trans Neural Netw 10:1032–1037
14. Matei O, Pop P, Vălean (2013) Optical character recognition in real environments using neural networks and k-nearest neighbor. Appl Intell 39:739–748
15. Owusu E, Zhan Y, Mao Q (2013) An SVM-AdaBoost facial expression recognition system. Appl Intell. doi:10.1007/s10489-013-0478-9
16. Peng X (2010) TSVR: an efficient twin support vector machine for regression. Neural Netw 23(3):365–372
17. Peng X (2010) A new twin support vector machine classifier and its geometric algorithms. Inf Sci 180(20):3863–3875
18. Ripley B (1996) Pattern recognition and neural networks. Cambridge University Press, Cambridge
19. Shao Y, Wang Z, Chen W, Deng N (2013) Least squares twin parametric-margin support vector machine for classification. Appl Intell 39:451–464
20. Vapnik V (1995) The nature of statistical learning theory. Springer, New York
21. Wang C, You W (2013) Boosting-SVM: effective learning with reduced data dimension. Appl Intell 39:465–474
22. Xu Y (2012) A rough margin-based linear ν support vector regression. Stat Probab Lett 82:528–534
23. Xu Y, Guo R (2014) An improved v-twin support vector machine. Applied Intell. doi:10.1007/s10489-013-0500-2
24. Xu Y, Wang L (2005) Fault diagnosis system based on rough set theory and support vector machine. Lect Notes Artif Intell 3614:981–988
25. Xu Y, Wang L (2012) A weighted twin support vector regression. Knowl-Based Syst 33:92–101
26. Xu Y, Wang L, Zhong P (2012) A rough margin-based ν-twin support vector machine. Neural Comput Appl 21:1307–1317
27. Xu Y, Lv X, Xi W (2012) A weighted multi-output support vector regression and its application. J Comput Inf Syst 8(9):3807–3814
28. Ye Q, Zhao C, Gao S, Zheng H (2012) Weighted twin support vector machines with local information and its application. Neural Netw 35:31–39

Yitian Xu is now an associate professor and supervisors for Ph.D candidates at College of Science in China Agricultural University. He received his Ph.D. in College of science from China Agricultural University in 2007. His research interests include machine learning and Data mining. He has published about 40 papers, and his research has appeared in Journal of Knowledge-based Systems, Neural Computing with Applications, Cognitive Computation, and so on. He is a visiting scholar at Department of Computer Science and Engineering in Arizona State University from August 2013 to August 2014.

Laisheng Wang received his Ph.D. in College of Economics Management from China Agricultural University in 2001. Dr. Wang is now a Professor of College of science, China Agricultural University. His research interests include machine learning, pattern recognition and data mining.